2. dplyr

데이터 프레임 다루기

패키지 dplyr

- tidyverse의 핵심 패키지
- 통계 데이터 세트: 변수가 열, 관찰값이 행을 이루는 2차원 구조. 데이터 프레임으로 입력
- 입력된 대부분의 데이터 프레임: 바로 분석할 수 있는 상태가 아님
  - 분석에 필요한 적절한 변수가 없음
  - 특정 조건을 만족하는 관찰값만을 선택
  - 관찰값의 순서 변경
- 데이터 프레임 다듬기
  - 매우 필요하나 시간이 많이 소요되는 힘든 작업
  - 일관된 법칙에 따라 편리하게 적용되는 기법이 절실하게 필요한 상황

패키지 dplyr의 주요 함수

- filter( ): 특정 조건을 만족하는 관찰값 선택
- arrange( ): 특정 변수를 기준으로 관찰값 정렬
- select( ): 변수 선택
- mutate( ): 새로운 변수 생성
- group_by( ): 그룹 생성
- summarize( ): 자료 요약

1. 조건에 따른 관찰값의 선택: filter()

- 특정 조건을 만족하는 관찰값 선택 가능

- 기본적인 형태: filter(df, condition)
  - df: 데이터 프레임
  - condition: 관찰값 선택 조건
    - 비교 연산자(>, >=, <, <=, !=, ==)
    - 논리 연산자(&, |, !)
    - 연산자 %in%

- 예제: mtcars
  - 변수 mpg가 30 이상인 자동차 선택
    - 함수 `filter()`: 전통적 데이터 프레임 입력 → 전통적 데이터 프레임 출력
      tibble 입력 → tibble 출력

```
> library(tidyverse)
> mtcars_t <- as.tibble(mtcars)
```

```
> filter(mtcars_t, mpg >= 30)
```

```
# A tibble: 4 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  32.4     4  78.7    66  4.08  2.2   19.5     1     1     4     1
2  30.4     4  75.7    52  4.93  1.62  18.5     1     1     4     2
3  33.9     4  71.1    65  4.22  1.84  19.9     1     1     4     1
4  30.4     4  95.1   113  3.77  1.51  16.9     1     1     5     2
```

- 변수 mpg가 30 이상이고, 변수 wt가 1.8 미만인 자동차 선택

```
> filter(mtcars_t, mpg >= 30 & wt < 1.8)
```

```
# A tibble: 2 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  30.4     4  75.7    52  4.93  1.62  18.5     1     1     4     2
2  30.4     4  95.1   113  3.77  1.51  16.9     1     1     5     2
```

- 변수 mpg가 30 이하, 변수 cyl이 6 또는 8, 변수 am이 1인 자동차 선택

```
> filter(mtcars_t, mpg <= 30, cyl %in% c(6,8), am == 1)
```

```
# A tibble: 5 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  21       6   160   110  3.9   2.62  16.5     0     1     4     4
2  21       6   160   110  3.9   2.88  17.0     0     1     4     4
3  15.8     8   351   264  4.22  3.17  14.5     0     1     5     4
4  19.7     6   145   175  3.62  2.77  15.5     0     1     5     6
5  15       8   301   335  3.54  3.57  14.6     0     1     5     8
```

- 논리 연산자 '&' 대신 콤마(,) 사용 가능
- cyl == 6 | cyl == 8 대신 cyl %in% c(6,8)

- 변수 mpg의 값이 mpg의 중앙값과 Q_3 사이에 있는 자동차 선택

  - 분위수 계산: 함수 quantile(x, probs=)

```
> filter(mtcars_t, mpg >= median(mpg) &
           mpg <= quantile(mpg, probs=0.75))
```

```
> filter(mtcars_t,between(mpg,median(mpg),
                          quantile(mpg,probs=0.75)))
```

  - 벡터 x가 특정 두 숫자(left, right) 사이에 있는지 확인
    1) x >= left & x <= right
    2) between(x, left, right)

```
# A tibble: 10 x 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21       6  160    110  3.9   2.62  16.5     0     1     4     4
 2  21       6  160    110  3.9   2.88  17.0     0     1     4     4
 3  22.8     4  108     93  3.85  2.32  18.6     1     1     4     1
 4  21.4     6  258    110  3.08  3.22  19.4     1     0     3     1
 5  22.8     4  141.    95  3.92  3.15  22.9     1     0     4     2
 6  19.2     6  168.   123  3.92  3.44  18.3     1     0     4     4
 7  21.5     4  120.    97  3.7   2.46  20.0     1     0     3     1
 8  19.2     8  400    175  3.08  3.84  17.0     0     0     3     2
 9  19.7     6  145    175  3.62  2.77  15.5     0     1     5     6
10  21.4     4  121    109  4.11  2.78  18.6     1     1     4     2
```

- 예제: airquality
  - 변수 Ozone 또는 Solar.R이 결측값인 관찰값 선택

```
> airs <- as_tibble(airquality)
> filter(airs, is.na(Ozone) | is.na(Solar.R))
```

```
# A tibble: 42 x 6
   Ozone Solar.R  Wind  Temp Month   Day
   <int>   <int> <dbl> <int> <int> <int>
 1    NA      NA  14.3    56     5     5
 2    28      NA  14.9    66     5     6
 3    NA     194   8.6    69     5    10
 4     7      NA   6.9    74     5    11
 5    NA      66  16.6    57     5    25
 6    NA     266  14.9    58     5    26
 7    NA      NA   8      57     5    27
 8    NA     286   8.6    78     6     1
 9    NA     287   9.7    74     6     2
10    NA     242  16.1    67     6     3
# ... with 32 more rows
```

2. 관찰값의 정렬: arrange()

- 특정 변수를 기준으로 데이터 프레임의 행 재배열

- 기본적인 사용법: arrange(df, var_1, var_2, …)
  - df: 데이터 프레임
  - var_1: 제1 정렬 기준 변수
  - var_2: var_1의 값이 같은 관찰값의 정렬 기준
  - 오름차순 정렬이 디폴트
  - 내림차순 정렬: 기준 변수를 함수 desc()에 입력

- 예제: mtcars
  - 변수 mpg의 값이 가장 좋지 않은 자동차부터 재배열

```
> mtcars_t <- as_tibble(mtcars)
> arrange(mtcars_t, mpg)
```

```
# A tibble: 32 x 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  10.4     8  472    205  2.93  5.25  18.0     0     0     3     4
 2  10.4     8  460    215  3     5.42  17.8     0     0     3     4
 3  13.3     8  350    245  3.73  3.84  15.4     0     0     3     4
 4  14.3     8  360    245  3.21  3.57  15.8     0     0     3     4
 5  14.7     8  440    230  3.23  5.34  17.4     0     0     3     4
 6  15       8  301    335  3.54  3.57  14.6     0     1     5     8
 7  15.2     8  276.   180  3.07  3.78  18       0     0     3     3
 8  15.2     8  304    150  3.15  3.44  17.3     0     0     3     2
 9  15.5     8  318    150  2.76  3.52  16.9     0     0     3     2
10  15.8     8  351    264  4.22  3.17  14.5     0     1     5     4
# ... with 22 more rows
```

- 변수 mpg가 가장 좋지 않은 자동차부터 배열하되, mpg 값이 같은 자동차는 변수 wt의 값이 높은 자동차부터 배열

```
> arrange(mtcars_t, mpg, desc(wt))
```

```
# A tibble: 32 x 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  10.4     8  460    215  3     5.42  17.8     0     0     3     4
 2  10.4     8  472    205  2.93  5.25  18.0     0     0     3     4
 3  13.3     8  350    245  3.73  3.84  15.4     0     0     3     4
 4  14.3     8  360    245  3.21  3.57  15.8     0     0     3     4
 5  14.7     8  440    230  3.23  5.34  17.4     0     0     3     4
 6  15       8  301    335  3.54  3.57  14.6     0     1     5     8
 7  15.2     8  276.   180  3.07  3.78  18       0     0     3     3
 8  15.2     8  304    150  3.15  3.44  17.3     0     0     3     2
 9  15.5     8  318    150  2.76  3.52  16.9     0     0     3     2
10  15.8     8  351    264  4.22  3.17  14.5     0     1     5     4
# ... with 22 more rows
```

- 예제: airquality

  - 5월 1일부터 5월 10일까지의 자료만을 대상으로 변수 Ozone의 값이 가장 낮았던 날부터 재배열

```
> airs <- as_tibble(airquality)
> airs_1 <- filter(airs, Month==5, Day<=10)
```

```
> arrange(airs_1, Ozone)
```

```
# A tibble: 10 x 6
   Ozone Solar.R  Wind  Temp Month   Day
   <int>   <int> <dbl> <int> <int> <int>
 1     8      19  20.1    61     5     9
 2    12     149  12.6    74     5     3
 3    18     313  11.5    62     5     4
 4    19      99  13.8    59     5     8
 5    23     299   8.6    65     5     7
 6    28      NA  14.9    66     5     6
 7    36     118   8      72     5     2
 8    41     190   7.4    67     5     1
 9    NA      NA  14.3    56     5     5
10    NA     194   8.6    69     5    10
```

- 데이터 프레임 airs_1을 변수 Ozone이 결측값인 케이스를 가장 앞으로 배열

```
> arrange(airs_1, !is.na(Ozone))
```

```
# A tibble: 10 x 6
   Ozone Solar.R  Wind  Temp Month   Day
   <int>   <int> <dbl> <int> <int> <int>
 1    NA      NA  14.3    56     5     5
 2    NA     194   8.6    69     5    10
 3    41     190   7.4    67     5     1
 4    36     118   8      72     5     2
 5    12     149  12.6    74     5     3
 6    18     313  11.5    62     5     4
 7    28      NA  14.9    66     5     6
 8    23     299   8.6    65     5     7
 9    19      99  13.8    59     5     8
10     8      19  20.1    61     5     9
```

원하는 형태의 배열이 된 이유

- 배열 기준으로 논리형 벡터 사용
- TRUE, FALSE 배열에서 우선 순위는 FALSE
- !is.na(Ozone): 변수 Ozone이 결측값인 케이스가 우선 순위

- 데이터 프레임 airs_1을 변수 Ozone이 가장 높은 날부터 배열하되 결측값이 있는 케이스를 가장 앞으로 배치

```
> arrange(airs_1, !is.na(Ozone), desc(Ozone))
```

```
# A tibble: 10 x 6
   Ozone Solar.R  Wind  Temp Month   Day
   <int>   <int> <dbl> <int> <int> <int>
 1    NA      NA  14.3    56     5     5
 2    NA     194   8.6    69     5    10
 3    41     190   7.4    67     5     1
 4    36     118   8      72     5     2
 5    28      NA  14.9    66     5     6
 6    23     299   8.6    65     5     7
 7    19      99  13.8    59     5     8
 8    18     313  11.5    62     5     4
 9    12     149  12.6    74     5     3
10     8      19  20.1    61     5     9
```

3. 변수의 선택: select()

- 데이터 세트의 크기 증가
- 변수의 개수가 수백 또는 수천이 되는 경우 발생
- 분석에 필요한 변수 선택으로 데이터 세트 크기 감소

- 기본적인 사용법: select(df, 변수 이름 또는 문자열 매칭 함수)
  - df: 데이터 프레임
  - 변수 이름 나열: 나열된 변수만 선택
  - 문자열 매칭 함수: 변수 선택을 효과적으로 할 수 있는 함수

- 예제: mtcars
  - row names를 변수 rowname으로 전환하고, 변수 rowname과 mpg 선택

```
> mtcars_t <- as_tibble(mtcars)
> mtcars_t <- rownames_to_column(mtcars_t, var="rowname")
```

```
> select(mtcars_t,rowname, mpg)
```

```
# A tibble: 32 x 2
   rowname             mpg
   <chr>             <dbl>
 1 Mazda RX4          21
 2 Mazda RX4 Wag      21
 3 Datsun 710         22.8
 4 Hornet 4 Drive     21.4
 5 Hornet Sportabout  18.7
 6 Valiant            18.1
 7 Duster 360         14.3
 8 Merc 240D          24.4
 9 Merc 230           22.8
10 Merc 280           19.2
# ... with 22 more rows
```

- 데이터 프레임 mtcars_t의 첫 번째 변수 mpg부터 여섯 번째 변수 wt까지 모두 선택

  - 연속된 순서의 변수 선택: 콜론 연산자 사용

```
> select(mtcars_t, mpg:wt)
```

```
# A tibble: 32 x 6
     mpg   cyl  disp    hp  drat    wt
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21        6  160    110  3.9   2.62
 2  21        6  160    110  3.9   2.88
 3  22.8      4  108     93  3.85  2.32
 4  21.4      6  258    110  3.08  3.22
 5  18.7      8  360    175  3.15  3.44
 6  18.1      6  225    105  2.76  3.46
 7  14.3      8  360    245  3.21  3.57
 8  24.4      4  147.    62  3.69  3.19
 9  22.8      4  141.    95  3.92  3.15
10  19.2      6  168.   123  3.92  3.44
# ... with 22 more rows
```

- 데이터 프레임 mtcars_t에서 변수 rowname과 qsec에서 carb까지 제거
  - 변수 제거: 마이너스 기호 사용

```
> select(mtcars_t, -rowname, -(qsec:carb))
```

```
# A tibble: 32 x 6
     mpg   cyl  disp    hp  drat    wt
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21        6  160    110  3.9   2.62
 2  21        6  160    110  3.9   2.88
 3  22.8      4  108     93  3.85  2.32
 4  21.4      6  258    110  3.08  3.22
 5  18.7      8  360    175  3.15  3.44
 6  18.1      6  225    105  2.76  3.46
 7  14.3      8  360    245  3.21  3.57
 8  24.4      4  147.    62  3.69  3.19
 9  22.8      4  141.    95  3.92  3.15
10  19.2      6  168.   123  3.92  3.44
# ... with 22 more rows
```

- 문자열 매칭 함수

  - 변수가 많은 경우 선택하고자 하는 변수의 이름을 일일이 나열하는 것은 상당히 비효율적

  - 효율적으로 변수 이름을 선택하는 방법이 필요

  - 문자열 매칭 함수

    `start_with("x")`: 이름이 "x"로 시작하는 변수 선택

    `ends_with("x")`: 이름이 "x"로 끝나는 변수 선택

    `contains("x")`: 이름에 "x"가 있는 변수 선택

    `num_range("x", 1:10)`: x1, x2, ..., x10과 동일

- 데이터 프레임 mtcars_t에서
  - "m"으로 시작되는 변수 선택
  - "p"로 이름이 끝나는 변수 선택
  - "a"가 이름에 있는 변수 선택

```
> select(mtcars_t, starts_with("m"))
```

```
> select(mtcars_t, ends_with("p"))
```

```
> select(mtcars_t, contains("a"))
```

```
# A tibble: 32 x 1
     mpg
   <dbl>
 1  21
 2  21
 3  22.8
 4  21.4
 5  18.7
 6  18.1
 7  14.3
 8  24.4
 9  22.8
10  19.2
# ... with 22 more rows
```

```
# A tibble: 32 x 2
    disp    hp
   <dbl> <dbl>
 1  160    110
 2  160    110
 3  108     93
 4  258    110
 5  360    175
 6  225    105
 7  360    245
 8  147.    62
 9  141.    95
10  168.   123
# ... with 22 more rows
```

```
# A tibble: 32 x 5
   rowname          drat    am  gear  carb
   <chr>           <dbl> <dbl> <dbl> <dbl>
 1 Mazda RX4        3.9     1     4     4
 2 Mazda RX4 Wag    3.9     1     4     4
 3 Datsun 710       3.85    1     4     1
 4 Hornet 4 Drive   3.08    0     3     1
 5 Hornet Sportabout 3.15   0     3     2
 6 Valiant          2.76    0     3     1
 7 Duster 360       3.21    0     3     4
 8 Merc 240D        3.69    0     4     2
 9 Merc 230         3.92    0     4     2
10 Merc 280         3.92    0     4     4
# ... with 22 more rows
```

- 예제: iris
  - 데이터 프레임 iris: 3종류의 붓꽃 각 50송이씩 150송이 붓꽃의 꽃받침 길이와 폭, 꽃잎의 길이와 폭을 측정한 데이터
  - 변수 이름

```
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
[5] "Species"
```

  - "pe"가 이름에 있는 변수 선택
  - "pe"가 이름에 있는 변수 제거

- "pe"가 이름에 있는 변수 선택

```
> iris <- as_tibble(iris)
> select(iris, contains("pe"))
```

```
# A tibble: 150 x 3
   Petal.Length Petal.Width Species
          <dbl>       <dbl> <fct>
 1          1.4         0.2 setosa
 2          1.4         0.2 setosa
 3          1.3         0.2 setosa
 4          1.5         0.2 setosa
 5          1.4         0.2 setosa
 6          1.7         0.4 setosa
 7          1.4         0.3 setosa
 8          1.5         0.2 setosa
 9          1.4         0.2 setosa
10          1.5         0.1 setosa
# ... with 140 more rows
```

- "pe"와 "Pe"가 모두 선택
- 소문자, 대문자 구분하지 않음
- 함수 starts_with( ), ends_with( ), contains( )에서 옵션 ignore.case=TRUE가 디폴트이기 때문

```
> select(iris, contains("pe", ignore.case=FALSE))
```

```
# A tibble: 150 x 1
   Species
   <fct>
 1 setosa
 2 setosa
 3 setosa
 4 setosa
 5 setosa
 6 setosa
 7 setosa
 8 setosa
 9 setosa
10 setosa
# ... with 140 more rows
```

- "pe"가 이름에 있는 변수 제거
  - 변수 제거: 마이너스 기호 사용

```
> select(iris,-contains("pe", ignore.case=FALSE))
```

```
# A tibble: 150 x 4
   Sepal.Length Sepal.Width Petal.Length Petal.Width
          <dbl>       <dbl>        <dbl>       <dbl>
 1          5.1         3.5          1.4         0.2
 2          4.9         3            1.4         0.2
 3          4.7         3.2          1.3         0.2
 4          4.6         3.1          1.5         0.2
 5          5           3.6          1.4         0.2
 6          5.4         3.9          1.7         0.4
 7          4.6         3.4          1.4         0.3
 8          5           3.4          1.5         0.2
 9          4.4         2.9          1.4         0.2
10          4.9         3.1          1.5         0.1
# ... with 140 more rows
```

- "Sp"로 이름이 시작되는 변수 제거
- "th"로 이름이 끝나는 변수 제거

```
> select(iris, -starts_with("Sp"))
```

```
> select(iris, -ends_with("th"))
```

```
# A tibble: 150 x 4
   Sepal.Length Sepal.Width Petal.Length Petal.Width
          <dbl>       <dbl>        <dbl>       <dbl>
 1          5.1         3.5          1.4         0.2
 2          4.9         3            1.4         0.2
 3          4.7         3.2          1.3         0.2
 4          4.6         3.1          1.5         0.2
 5          5           3.6          1.4         0.2
 6          5.4         3.9          1.7         0.4
 7          4.6         3.4          1.4         0.3
 8          5           3.4          1.5         0.2
 9          4.4         2.9          1.4         0.2
10          4.9         3.1          1.5         0.1
# ... with 140 more rows
```

```
# A tibble: 150 x 1
   Species
   <fct>
 1 setosa
 2 setosa
 3 setosa
 4 setosa
 5 setosa
 6 setosa
 7 setosa
 8 setosa
 9 setosa
10 setosa
# ... with 140 more rows
```

- 변수 배열 변경: 몇몇 변수를 제일 앞으로 옮겨서 다시 배치

  데이터 프레임 iris에서 마지막 변수 Species를 첫 번째 변수로 재배열

  - 함수 everything()

```
> select(iris, Species, everything())
```

```
# A tibble: 150 x 5
   Species Sepal.Length Sepal.Width Petal.Length Petal.Width
   <fct>          <dbl>       <dbl>        <dbl>       <dbl>
 1 setosa           5.1         3.5          1.4         0.2
 2 setosa           4.9         3            1.4         0.2
 3 setosa           4.7         3.2          1.3         0.2
 4 setosa           4.6         3.1          1.5         0.2
 5 setosa           5           3.6          1.4         0.2
 6 setosa           5.4         3.9          1.7         0.4
 7 setosa           4.6         3.4          1.4         0.3
 8 setosa           5           3.4          1.5         0.2
 9 setosa           4.4         2.9          1.4         0.2
10 setosa           4.9         3.1          1.5         0.1
# ... with 140 more rows
```

- 변수 이름 수정

  - 함수 select(): 이름이 명시되지 않는 변수는 자동 제거
  - 함수 rename(): 변수 이름 수정에는 더 효과적

  - 데이터 프레임 mtcars_t에서 변수 rowname을 Model로 이름 수정

```
> select(mtcars_t, Model=rowname)
```

```
# A tibble: 32 x 1
   Model
   <chr>
 1 Mazda RX4
 2 Mazda RX4 Wag
 3 Datsun 710
 4 Hornet 4 Drive
 5 Hornet Sportabout
 6 Valiant
 7 Duster 360
 8 Merc 240D
 9 Merc 230
10 Merc 280
# ... with 22 more rows
```

변수 Model만 남아 있음

```
> rename(mtcars_t, Model=rowname)
```

```
# A tibble: 32 x 12
   Model          MPG   cyl  disp    hp  drat    wt  qsec    vs    am  gear
   <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 Mazda RX4     21       6  160    110  3.9   2.62  16.5     0     1     4
 2 Mazda RX4~    21       6  160    110  3.9   2.88  17.0     0     1     4
 3 Datsun 710    22.8     4  108     93  3.85  2.32  18.6     1     1     4
 4 Hornet 4 ~    21.4     6  258    110  3.08  3.22  19.4     1     0     3
 5 Hornet Sp~    18.7     8  360    175  3.15  3.44  17.0     0     0     3
 6 Valiant       18.1     6  225    105  2.76  3.46  20.2     1     0     3
 7 Duster 360    14.3     8  360    245  3.21  3.57  15.8     0     0     3
 8 Merc 240D     24.4     4  147.    62  3.69  3.19  20       1     0     4
 9 Merc 230      22.8     4  141.    95  3.92  3.15  22.9     1     0     4
10 Merc 280      19.2     6  168.   123  3.92  3.44  18.3     1     0     4
# ... with 22 more rows, and 1 more variable: carb <dbl>
```

다른 모든 변수가 남아 있음

4. 새로운 변수의 추가: mutate()

- 데이터 프레임을 구성하고 있는 변수를 이용하여 새로운 변수 생성

- 기본적인 사용법: mutate(df, 새로운 변수 생성 표현식)
  - df: 데이터 프레임
  - 다수의 변수 생성 표현식: 콤마로 구분하여 나열

- 생성된 변수는 데이터 프레임의 마지막 변수로 추가됨

- 예제: mtcars_t
  - 변수 kml과 gp_kml 생성하고 데이터 프레임의 처음 두 변수로 추가
    kml: 1 mpg는 0.43 kml
    gp_kml: kml이 10 이상이면 'good', 10 미만이면 'bad'

```
> mtcars_t <- as_tibble(mtcars)
> mtcars_t <- rownames_to_column(mtcars_t, var="Model")
```

```
> mtcars_t <- mutate(mtcars_t,
                     kml=mpg*0.43,
                     gp_kml=if_else(kml>=10,"good","bad"))
```

- if_else( ): ifelse( )와 기본적인 사용법은 동일
- if_else(condition, true, false) : true와 false의 유형이 같아야 함
  - ifelse(kml>=10, 1, "2"): 문제 없음
  - if_else(kml>=10, 1, "2"): 오류

```
> select(mtcars_t, kml, gp_kml, everything())
```

```
# A tibble: 32 x 14
     kml gp_kml Model       mpg   cyl  disp    hp  drat    wt  qsec    vs
   <dbl> <chr>  <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  9.03 bad    Mazda RX4  21       6  160    110  3.9   2.62  16.5     0
 2  9.03 bad    Mazda RX~  21       6  160    110  3.9   2.88  17.0     0
 3  9.80 bad    Datsun 7~  22.8     4  108     93  3.85  2.32  18.6     1
 4  9.20 bad    Hornet 4~  21.4     6  258    110  3.08  3.22  19.4     1
 5  8.04 bad    Hornet S~  18.7     8  360    175  3.15  3.44  17.0     0
 6  7.78 bad    Valiant    18.1     6  225    105  2.76  3.46  20.2     1
 7  6.15 bad    Duster 3~  14.3     8  360    245  3.21  3.57  15.8     0
 8 10.5  good   Merc 240D  24.4     4  147.    62  3.69  3.19  20       1
 9  9.80 bad    Merc 230   22.8     4  141.    95  3.92  3.15  22.9     1
10  8.26 bad    Merc 280   19.2     6  168.   123  3.92  3.44  18.3     1
# ... with 22 more rows, and 3 more variables: am <dbl>, gear <dbl>,
#   carb <dbl>
```

- 새로운 변수만 유지하고 나머지 변수 모두 삭제
  - 함수 transmute() 사용

```
> transmute(mtcars_t,
            kml=mpg*0.43,
            gp_kml=if_else(kml>=10,"good","bad"))
```

```
# A tibble: 32 x 2
     kml gp_kml
   <dbl> <chr>
 1  9.03 bad
 2  9.03 bad
 3  9.80 bad
 4  9.20 bad
 5  8.04 bad
 6  7.78 bad
 7  6.15 bad
 8 10.5  good
 9  9.80 bad
10  8.26 bad
# ... with 22 more rows
```

5. 그룹 생성 및 그룹별 자료 요약: group_by, summarize()

- 함수 `summarize()`: 변수의 요약 통계량 계산

- 기본적인 사용법: `summarize(df, name=fun)`
  - 결과: tibble
  - name: 계산된 통계량 값의 이름

- 데이터 프레임 mpg의 변수 hwy의 평균 계산

```
> summarize(mpg, hwy_mpg=mean(hwy))
# A tibble: 1 x 1
  hwy_mpg
    <dbl>
1    23.4
```

- 함수 summarize()에서 사용되는 함수
  - 결과가 숫자 하나로 출력되는 함수만 사용 가능: mean(), sd(), min(), max()
  - 결과가 벡터인 함수는 사용 불가. 예) range()
  - 유용한 함수:
    - n(): 케이스의 개수
    - n_distinct(): 서로 다른 숫자의 개수

```
> summarize(mpg, n=n(), n_hwy=n_distinct(hwy),
            avg_hwy=mean(hwy), sd_hwy=sd(hwy))
# A tibble: 1 x 4
      n n_hwy avg_hwy sd_hwy
  <int> <int>   <dbl>  <dbl>
1   234    27    23.4   5.95
```

  - 함수 summarize()는 그 자체만으로는 그렇게 특별히 유용한 함수는 아님
  - 함수 group_by()와 함께 사용되면 매우 강력한 분석 도구가 됨

- 함수 group_by()
  - 한 개 이상의 변수로 데이터 프레임을 그룹으로 구분
  - 기본적인 사용법: group_by(df, var)
  - 실행 결과: gouped_df 라는 class 속성이 추가된 tibble. 출력된 상태로는 실행 전과 차이가 없음.

- 데이터 프레임 mpg를 변수 cyl에 따라 그룹으로 구분하고, 각 그룹에 속한 케이스의 개수 및 각 그룹별 변수 hwy의 평균값 계산

```
> by_cyl <- group_by(mpg, cyl)
> summarize(by_cyl, n=n(), avg_mpg=mean(hwy))
# A tibble: 4 x 3
    cyl     n avg_mpg
  <int> <int>   <dbl>
1     4    81    28.8
2     5     4    28.8
3     6    79    22.8
4     8    70    17.6
```

- by_cyl: 분석 중간 단계에서 생성되는 객체
- 자체로는 큰 의미가 없음
- 복잡한 분석에서는 무수히 발생
- pipe 기능 필요

6. Pipe 기능

- pipe: 한 명령문의 결과물을 바로 다음 명령문의 입력 요소로 직접 사용할 수 있도록 명령문을 서로 연결하는 기능

- pipe 연산자: %>%
  - 기본적인 사용법: `lhs %>% rhs`
  - 예:
    - `x %>% f` → `f(x)`
    - `x %>% f(y)` → `f(x, y)` : 첫 번째 요소
    - `x %>% f(y, .)` → `f(y, x)`: 첫 번째 요소가 아닌 경우에는 해당 위치에 마침표

- 데이터 프레임 mpg를 변수 cyl에 따라 그룹으로 구분하고, 각 그룹에 속한 케이스의 개수 및 각 그룹별 변수 hwy의 평균값 계산
- pipe 기능을 사용하여 계산

```
> mpg %>%
    group_by(cyl) %>%
    summarize(n=n(), avg_mpg=mean(hwy))
```

```
# A tibble: 4 x 3
    cyl     n avg_mpg
  <int> <int>   <dbl>
1     4    81    28.8
2     5     4    28.8
3     6    79    22.8
4     8    70    17.6
```

- 예제: airquality

  pipe 기능을 사용하여 다음을 구하라.

  1) 월별 변수 Ozone의 평균값
  2) 월별 날수, 변수 Ozone에 결측값이 있는 날수 및 실제 측정이 된 날수
  3) 월별 첫날과 마지막 날 변수 Ozone의 값
  4) 월별 변수 Ozone의 가장 작은 값과 가장 큰 값
  5) 월별로 변수 Ozone의 개별 값이 전체 기간 동안의 평균값보다 작은 날수

  - 다섯 문제 모두 월별로 구분된 데이터 프레임 필요

```
> airs_Month <- airquality %>%
    group_by(Month)
```

- 월별 변수 Ozone의 평균값

```
> airs_Month %>%
    summarize(avg_Oz=mean(Ozone,na.rm=TRUE))
```

```
# A tibble: 5 x 2
  Month avg_Oz
  <int>  <dbl>
1     5   23.6
2     6   29.4
3     7   59.1
4     8   60.0
5     9   31.4
```

- 월별 날수, 변수 Ozone에 결측값이 있는 날수 및 실제 측정이 된 날수

```
> airs_Month %>%
    summarize(n_total=n(),n_miss=sum(is.na(Ozone)),
              n_obs=sum(!is.na(Ozone)))
```

```
# A tibble: 5 x 4
  Month n_total n_miss n_obs
  <int>   <int>  <int> <int>
1     5      31      5    26
2     6      30     21     9
3     7      31      5    26
4     8      31      5    26
5     9      30      1    29
```

- 월별 첫날과 마지막 날 변수 Ozone의 값

  - 필요한 함수: first( ), last( ), nth( )
    - first(x): x[1]
    - last(x): x[length(x)]
    - nth(x,2): x[2]

  - 부가 기능: 옵션 order_by=, default=

order_by=

```
> x <- c(2,4,6,8,10)
> first(x)
[1] 2
> first(x, order_by=order(-x))
[1] 10
```

- 옵션 order_by=에 지정된 벡터로 순서 결정하고, 해당 위치에 있는 벡터 x의 값 출력
- order(x) vs order(-x)

default=

```
> nth(x,2)
[1] 4
> nth(x,-2)
[1] 8
> nth(x,6)
[1] NA
> nth(x,6,default=99)
[1] 99
```

- nth(x,2): 앞에서 두 번째 자료
- nth(x, -2): 끝에서 두 번째 자료
- 찾는 위치에 해당하는 자료가 없으면 NA
- 찾는 위치에 해당하는 자료가 없는 경우 default에 지정된 값 출력

- 월별 첫날과 마지막 날 변수 Ozone의 값

```
> airs_Month %>%
    summarise(first_Oz=first(Ozone), last_Oz=last(Ozone))
```

```
# A tibble: 5 x 3
  Month first_Oz last_Oz
  <int>    <int>   <int>
1     5       41      37
2     6       NA      NA
3     7      135      59
4     8       39      85
5     9       96      20
```

- 월별 변수 Ozone의 가장 작은 값과 가장 큰 값

```
> airs_Month %>%
    summarize(max_Oz=max(Ozone,na.rm=TRUE),
              min_Oz=min(Ozone,na.rm=TRUE))
```

```
# A tibble: 5 x 3
  Month max_Oz min_Oz
  <int>  <dbl>  <dbl>
1     5    115      1
2     6     71     12
3     7    135      7
4     8    168      9
5     9     96      7
```

- 월별로 변수 Ozone의 개별 값이 전체 기간 동안의 평균값보다 작은 날수

  - 변수 Ozone의 전체 기간 동안의 평균값 계산

```
> m_oz <- airs_Month %>%
    with(mean(Ozone, na.rm=TRUE))
> m_oz
[1] 42.12931
```

  - 개별 값이 m_oz 보다 작은 날수

```
> airs_Month %>%
    summarize(low_Oz=sum(Ozone<m_Oz,na.rm=TRUE))
```

```
# A tibble: 5 x 2
  Month low_Oz
  <int>  <int>
1     5     24
2     6      8
3     7      8
4     8     10
5     9     22
```